To appear in *Advanced Robotics*
Vol. 00, No. 00, January 2013, 1–13

FULL PAPER

Guaranteed Interval Analysis Localization for Mobile Robots

Rémy Guyonneau*, Sébastien Lagrange, Laurent Hardouin and Philippe Lucidarme

Laboratoire d'Ingénierie des Systèmes Automatisés (LISA), Université d'Angers, 62 avenue Notre Dame du Lac, Angers, France

(*v1.0 released January 2013*)

This paper presents a set membership method (named Interval Analysis Localization) to deal with the global localization problem of mobile robots. By using a range sensor, the odometry and a discrete map of an indoor environment, a robot has to determine its pose (position and orientation) in the map without any knowledge of its initial pose. In a bounded error context the IAL (Interval Analysis Localization) algorithm searches a box (interval vector) as small as possible that includes the robot's pose. The localization process is based on interval analysis and constraint propagation tools. The proposed method is validated using real data recorded during the CAROTTE challenge, organized by the French ANR (National Research Agency) and the French DGA (General Delegation of Armament). Interval Analysis Localization is then compared with the well known Monte Carlo Localization showing weaknesses and strengths of both algorithms. As it is shown in this paper with the IAL algorithm, interval analysis can be an efficient tool to solve the global localization problem.

Keywords: interval analysis; bounded errors; global localization; mobile robots; kidnapping

1. Introduction

Robot localization is one of the most important issue of mobile robotics [1, 2]: a robot has to know its location to be able to perform navigation tasks. The localization problem can be divided into two categories: the pose tracking and the global localization. In the pose tracking problem a robot has to find its new pose using the knowledge of its initial pose. Usually this is done in real time while the robot is exploring its environment. In the global localization problem a robot has to find its pose without the knowledge of its initial pose.

Most of the proposed solutions to localize a robot are based on probabilistic estimation techniques (see [3, 4]). The Kalman Filter [5, 6] and its improvements [7] are used for the pose tracking problem [8] and more precisely for the SLAM problem (see [9, 10]). Particle filters [11] with for example the Monte Carlo algorithm [12] and its spin-off [13, 14] are used to deal with the global localization problem.

In this paper a set membership approach, named **I**nterval **A**nalysis **L**ocalization (IAL) algorithm, will be considered for the global localization problem and compared to a **M**onte **C**arlo **L**ocalization (MCL) algorithm. Set membership localization algorithms already exist. Using a **S**et **I**nversion **V**ia **I**nterval **A**nalysis (SIVIA) approach [15] and a environment vectorization [16] for example, or combining stochastic and set membership tools [17].

IAL algorithm considers a map, obtained for example by using classical SLAM techniques, and do not need environment vectorization, unlike [16]. The *LORIA* and *CAROTTE* arena maps considered later in Section 4 have been entirely provided by actual robots: they are sets of points

*Corresponding author. Email: remy.guyonneau@univ-angers.fr

and did not have any human operator modification (as it can be done in [15] to notify seamarks). Furthermore it can be noticed that no landmarks are considered.

The presented method is a full deterministic and repeatable set membership method that does not consider any stochastic tool, unlike [17]. Thus the results of the localization process are obtained in a guaranteed way, according to a fixed number of outliers (measurements that are not consistent with the problem). Furthermore the IAL algorithm is able to deal with the kidnapping problem (a robot is moved when performing a pose tracking without knowing it).

The proposed algorithm combined two interval tools: contraction and bisection. First the localization problem is seen as a **C**onstraint **S**atisfaction **P**roblem (CSP). Using contractors, the set of feasible poses is reduced (contracted). Then when the CSP failed to contract the research domain, a bisection is performed using a SIVIA algorithm. The combination of those two methods is more efficient than just a SIVIA, as it can be done in [16] (see Section 4.1).

The paper is organized as follows. First the considered global localization problem is presented in Section 2. The IAL algorithm is detailed in Section 3 and experimental results are presented in Section 4. Finally Section 5 compares the IAL and the MCL algorithms and Section 6 concludes this paper.

2. The Global Localization Problem

This section presents the considered localization problem.

2.1 *The Robot*

A mobile wheeled robot (depicted in Figure 2.1) with a range sensor is considered. This system is characterized by the following discrete time dynamic equations:

$$\mathbf{q}(k+1) = f(\mathbf{q}(k), \mathbf{u}(k)), \tag{1}$$

$$\mathbf{y}(k) = g_{\varepsilon}(\mathbf{q}(k)). \tag{2}$$

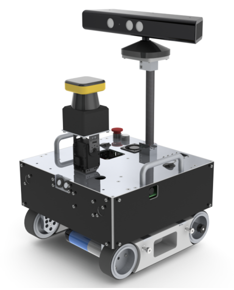


Figure 1. Robot used during the CAROTTE challenge.

The robot's pose $\mathbf{q}(k) = (x_1(k), x_2(k), \theta(k))$ is defined by its location $\mathbf{x}(k) = (x_1(k), x_2(k))$ and its orientation $\theta(k)$ in the environment denoted ε at discrete time k. The environment $\varepsilon \in \mathbb{R}^2$ is a two dimensional domain where the robot moves. The function f characterizes the robot's dynamic and the vector $\mathbf{u}(k)$ corresponds to the control vector at time k.

Here is the dynamic function we have considered for our experimentations:

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ \theta(k+1) \end{pmatrix} = \begin{pmatrix} x_1(k) \\ x_2(k) \\ \theta(k) \end{pmatrix} + \begin{pmatrix} \sin(\theta(k)) \\ \cos(\theta(k)) \\ 1 \end{pmatrix} \cdot \begin{pmatrix} \Delta_{x,k} \\ \Delta_{x,k} \\ \Delta_{\theta,k} \end{pmatrix}, \tag{3}$$

with $\Delta_{x,k}$ the translation done by the robot between the time k and $k+1$, and $\Delta_{\theta,k}$ the rotation done by the robot between the time k and $k+1$. Those two values are estimated by the odometry's sensors.

The vector $\mathbf{y}(k) = (y_1(k), ..., y_n(k))$ is the vector of measurements. Note that $\mathbf{y}(k)$ depends on the robot pose $\mathbf{q}(k)$ and the environment ε. In fact, a measurement y_i corresponds to the distance into the direction γ_i between the robot and the first obstacle in ε (Figure 2).

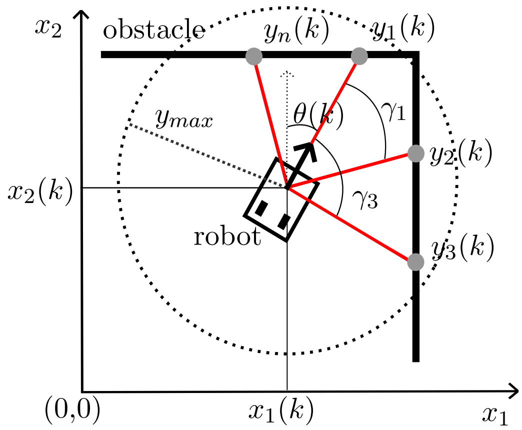


Figure 2. Sensor measurements $\mathbf{y}(k) = (y_1(k), ..., y_n(k))$. y_{max} denotes the maximal range of the sensor.

2.2 *The Environment*

Figure 3. The environment of the CAROTTE challenge.

The environment ε where the robot is moving is approximated by an occupancy grid map [4]. Figure 2.2 depicts an example of indoor environment. The grid map, named $\mathbb{G}$, is composed of $n \times m$ cells (i, j) and at each cell (i, j) is associated $g_{i,j} \in \{0, 1\}$:

$$g_{i,j} = \begin{cases} 0 & \text{if the cell corresponds to an obstacle-free subspace of } \varepsilon, \\ 1 & \text{else.} \end{cases} \quad (4)$$

$\mathbb{G}$ is a discrete version of ε. Figure 4 represents an example of occupancy grid map with 35×38 cells.

2.3 *The Objective*

The objective of the global localization problem is to find $\mathbf{q}(k)$ the pose of the robot at discrete time k, without any information about the initial pose $\mathbf{q}(0)$. This is done by using the sensor data $\mathbf{y}(0), ..., \mathbf{y}(k)$, the control data $\mathbf{u}(0), ..., \mathbf{u}(k-1)$ (estimated by the odometry), and the grid map $\mathbb{G}$.

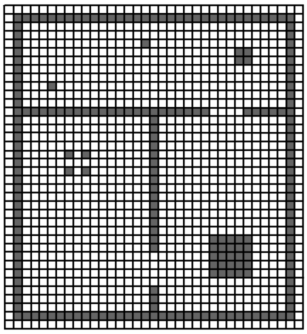


Figure 4. Example of occupancy grid map. White cells represent free spaces.

3. Interval Analysis Localization, a Deterministic Approach

The proposed method uses interval analysis and Constraint Satisfaction Problem (CSP) tools to solve the global localization problem. This section presents basics of those tools and then presents the IAL algorithm.

3.1 *Interval Analysis Introduction*

An *interval vector* [18], or a *box* $[\mathbf{q}]$ is defined as a closed subset of $\mathbb{R}^n$:$[\mathbf{q}] = ([x_1], [x_2], \cdots) = ([\underline{x_1}, \overline{x_1}], [\underline{x_2}, \overline{x_2}], \cdots)$.

The size of a box is defined as

$$size([\mathbf{q}]) = (\overline{x_1} - \underline{x_1}) \times (\overline{x_2} - \underline{x_2}) \times \cdots,$$

For instance $size([2, 5], [1, 8], [0, 5]) = 105$.

It can be noticed that any arithmetic operators such as $+, -, \times, \div$ and functions such as $exp, sin, sqr, sqrt, ...$ can be easily extended to intervals, [19].

A Constraint Satisfaction Problem (CSP) is defined by three sets. A set of variables $\mathcal{V}$, a set of domains $\mathcal{D}$ for those variables and a set of constraints $\mathcal{C}$ connecting the variables together.

Example of CSP:

$$\left\{ \begin{array}{c} \mathcal{V} = \{x_1, x_2, x_3\} \\ \mathcal{D} = \{x_1 \in [7, +\infty], x_2 \in [-\infty, 2], x_3 \in [-\infty, 9]\} \\ \mathcal{C} = \{x_1 = x_2 + x_3\} \end{array} \right\}. \tag{5}$$

Solving a CSP consists into reducing the domains by removing the values that are not consistent with the constraints. It can be efficiently solved by considering interval arithmetic [20]. For the previous example:

$$\begin{aligned} x_1 = x_2 + x_3 &\Rightarrow x_1 \in [x_1] \cap ([-\infty, 2] + [-\infty, 9]) \\ &\Rightarrow x_1 \in [7, +\infty] \cap [-\infty, 11] = [7, 11] \\ x_2 = x_1 - x_3 &\Rightarrow x_2 \in [x_2] \cap ([7, 11] - [-\infty, 9]) \\ &\Rightarrow x_2 \in [-\infty, 2] \cap [-2, +\infty] = [-2, 2] \\ x_3 = x_1 - x_2 &\Rightarrow x_3 \in [x_3] \cap ([7, 11] - [-2, 2]) \\ &\Rightarrow x_3 \in [-\infty, 2] \cap [5, 13] = [5, 13] \end{aligned}$$

The solutions of that CSP are the following contracted domains $[x_1]^* = [7, 11], [x_2]^* = [-2, 2]$ and $[x_3]^* = [5, 13]$.

Later the localization problem will be expressed as a CSP.

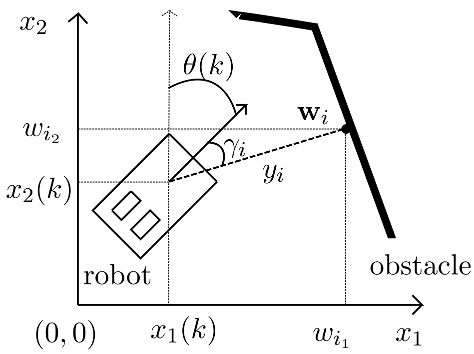


Figure 5. $\mathbf{w}_i$, the coordinates of the obstacle detected by y_i in the map's frame.

3.2 *The IAL algorithm*

Algorithm 1: Interval Analysis Localization

Data: $\mathcal{L}_{k-1}, [\mathbf{y}(k)], [\mathbf{u}(k-1)]$

1 $\mathcal{L}_k = \emptyset$;
2 **while** $\mathcal{L}_{k-1} \neq \emptyset$ **do**
3 $[\mathbf{q}(k-1)] = \mathcal{L}_{k-1}.pop\_back()$;
4 update $[\mathbf{q}(k-1)]$ to $[\mathbf{q}(k)]$ according to $[\mathbf{u}(k-1)]$;
5 contract $[\mathbf{q}(k)]$ by using $[\mathbf{y}(k)]$ and $\mathbb{G}$;
6 **if** $size([\mathbf{q}(k)]) > \xi$ **then**
7 bisect $[\mathbf{q}(k)]$ into $[\mathbf{q}_1(k)]$ and $[\mathbf{q}_2(k)]$;
8 $\mathcal{L}_k.push\_back([\mathbf{q}_1(k)])$;
9 $\mathcal{L}_k.push\_back([\mathbf{q}_2(k)])$;
10 **else**
11 **if** $[\mathbf{q}(k)] \neq \emptyset$ **then**
12 $\mathcal{L}_k.push\_back([\mathbf{q}(k)])$;

Result: $\mathcal{L}_k$.

A bounded error context is considered for the global localization problem presented in the Section 2. Thus an interval $[y_i(k)]$ can be associated to each measurement $y_i(k)$, according to the sensor accuracy, and an interval $[\mathbf{u}(k)]$ can be associated to $\mathbf{u}(k)$, such that $y_i(k) \in [y_i(k)]$ and $\mathbf{u}(k) \in [\mathbf{u}(k)]$.

In this context, the IAL algorithm is able to find a list $\mathcal{L}_k = \{[\mathbf{q}(k)]\}$, with $[\mathbf{q}(k)] = ([x_1(k)], [x_2(k)], [\theta(k)])$, that includes the robot pose $\mathbf{q}(k)$, $\mathbf{q}(k) \in \mathcal{L}_k$.

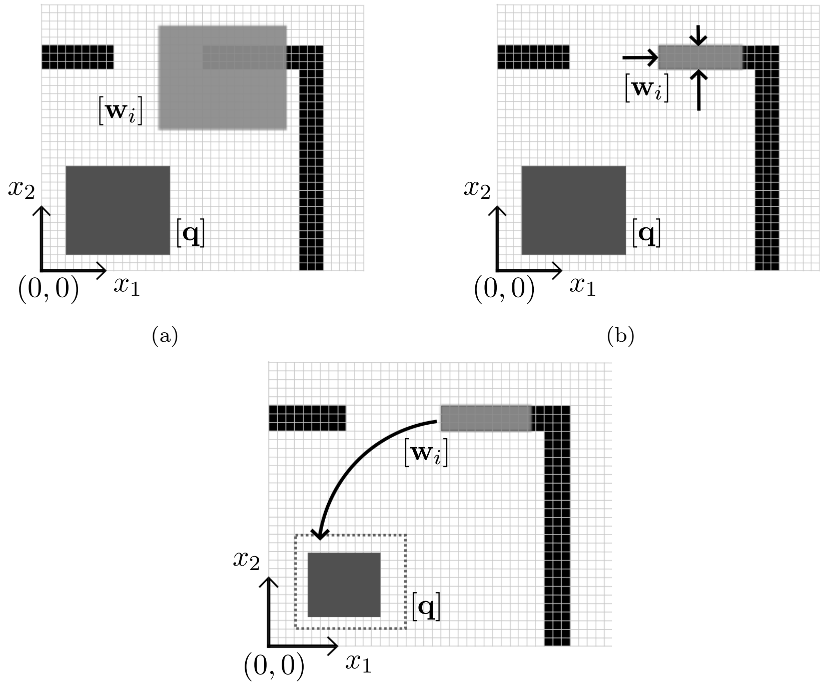
The IAL algorithm (presented Algorithm 1) has three important steps.

First, line 4, the prediction step. The pose at time k is evaluated from the pose at time $k-1$ by using Equation (1). The computation $[\mathbf{q}(k)] = f([\mathbf{q}(k-1)], [\mathbf{u}(k-1)])$ is done using interval arithmetic.

Then, line 5 corresponds to the contraction step. The measurements $\mathbf{y}(k)$ and the observation function g_ε are used to contract the boxes (pose estimations): the idea is to find $\{\mathbf{q}(k) | g_\varepsilon(\mathbf{q}(k)) = \mathbf{y}(k)\}$.

As ε is approximated by $\mathbb{G}$, IAL algorithm searches $\{[\mathbf{q}(k)] | g_{\mathbb{G}}([\mathbf{q}(k)]) = [\mathbf{y}(k)]\}$. This problem is seen as a CSP with the variables $\mathbf{q}(k), \mathbf{y}(k)$, the domains $[\mathbf{q}(k)], [\mathbf{y}(k)]$ and constraints built with the measurements $\mathbf{y}(k)$ and the map $\mathbb{G}$. To be compared with the map, the measurements $[y_i]$ are converted to obstacle coordinates $[\mathbf{w}_i] = ([w_{i_1}], [w_{i_2}])$ in $\mathbb{G}$'s frame, according to $[\mathbf{q}(k)]$ and γ_i, as depicted in Figure 5. An example of contraction using one measurement is presented in Figure 6.

Finally the last step of the IAL is the bisection line 7. If a contracted box $[\mathbf{q}(k)]$ is bigger than the minimal size ξ, $[\mathbf{q}(k)]$ is bisected into two boxes $[\mathbf{q}_1(k)]$ and $[\mathbf{q}_2(k)]$ such as $size([\mathbf{q}_1(k)]) =$



(c)

Figure 6. Example of contraction. In Figure 6(a) there is a grid map $\mathbb{G}$ (each black cell value is 1). The box $[\mathbf{w}_i]$ (light grey) is a guaranteed evaluation of the measurement $\mathbf{w}_i$ according to $[\mathbf{q}]$ (dark grey) and $[y_i]$. By using the constraint *a measurement has to intersect with an obstacle in the map* the domain $[\mathbf{w}_i]$ is contracted (Figure 6(b)) and then by using an other constraint (*the distance between the robot and the detected obstacle is* $y_i \in [y_i]$) it is possible to contract the domain $[\mathbf{q}]$ (Figure 6(c)).

$size([\mathbf{q}_2(k)])$ and $[\mathbf{q}'_1(k)] \cup [\mathbf{q}_2(k)] = [\mathbf{q}(k)]$.

As it can be noticed in line 11, bisected boxes that are not consistent with the localization problem are removed from the final solution. Note that outliers are handled using relaxed intersection [21]. Considering n_o outliers over n measurements, the contraction is done assuming that at least $n - n_o$ measurements have to intersect an obstacle in the map.

A kidnapping situation is detected when the resulting list of the algorithm $\mathcal{L}_k$ is empty (too many measurements are not consistent with the current pose estimation $\mathcal{L}_{k-1}$ and the map). When this occurs, the algorithm has to be initialized with all the possible poses (restart a global localization process over the entire environment).

4. Experimental Results

In the following the IAL algorithm is validated with experimentations. First the method is tested in a simulated environment and then considering two real situations.

The computer used to process those experimentations has the following specifications:

Memory: 2,0 GiB,
Processors: Intel(R) Core(TM) CPU 6420 @ 2.13GHz.

4.1 *A simulated arena*

In this Section the IAL algorithm is tested in a 19m × 3m environment (Figure 7). In order to compare those results with the method presented in [16], the following parameters are chosen:

- 20% of the measurements are outliers,
- The initial box size is 19m × 2m × 180 deg,

- The measurements have a distance error of ± 5cm and an angular error of ± 1 deg,
- Between each iteration the robot does a 1m move with an error of $\pm 10\%$.

The considered map and initial conditions are presented in Figure 7(a). The environment corresponds to a $19m$ corridor. In this example, six iterations (an iteration corresponds to a moving and a contraction) of the IAL algorithm are necessary to localize the robot. Five of those iterations are presented in Figure 7. The resulting box at time $k = 6$ is $[\mathbf{q}(6)] = ([555.36, 579.8]\text{cm}, [198.3, 204.25]\text{cm}, [267.9, 274.42]\text{ deg})$.

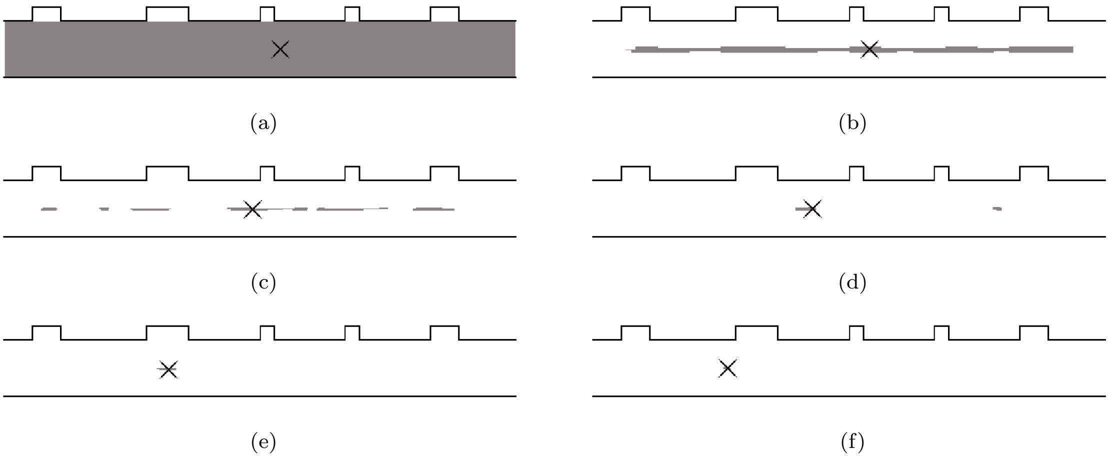


Figure 7. Figure 7(a) corresponds to the initial configuration of the localization problem. The cross represents the robot and the grey box the knowledge of its initial position $[\mathbf{q}(0)]$. Figure 7(b) corresponds to the first iteration of the IAL algorithm (the first contraction step). Figures 7(c), 7(d), 7(e) and 7(f) correspond respectively to the 2^{nd}, 3^{rd}, 5^{th} and 6^{th} iterations.

Table 1 presents the computation times of the six iterations. In [16], for a similar localization problem, the first iteration takes $6s$ and the last iterations are about $1s$ of execution time. It can be concluded that adding a contraction step in addition to the bisection improves significantly the computation time. Note that in Table 1 are indicated corresponding figures of Figure 7.

Table 1. The iteration computation times

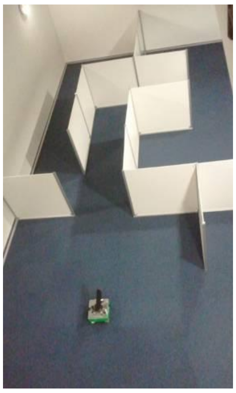
Iteration (Figure)	1 (Fig. 7(b))	2 (Fig. 7(c))	3 (Fig. 7(d))	4	5 (Fig. 7(e))	6 (Fig. 7(f))
Execution time (s)	2.76	1.15	0.24	0.05	0.06	0.03

4.2 *The LORIA arena*

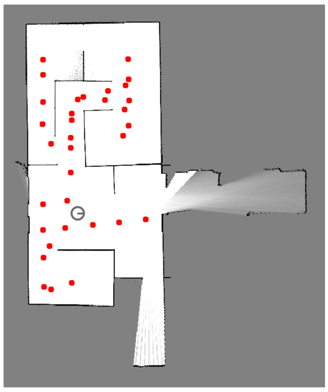
An actual 10m$\times$4.7m indoor environment is considered (Figure 8(a)). By using SLAM techniques a map of this environment is built, Figure 8(b). The objective of the experimentation is to test the IAL algorithm in a real context: map with imperfections and non filtered data provided by an actual sensor. Note that the map has 2cm $\times$ 2cm grid cells.

Static global localizations are performed at 33 different locations all over the environment, Figure 8(b). Note that for those experimentations, the robot is localised without moving. The measurement accuracy is assumed to be ± 1cm, i.e, to each measurement $y_i(k)$ the following interval is associated $[y_i(k) - 1\text{cm}, y_i(k) + 1\text{cm}]$. The results of those global localizations are summarised in Table 2.

The execution time evolves between 21 and 45.5 seconds. This is due to the first iteration of the algorithm which starts with all the environment as possible position and all the possible orientation ($[0, 2\pi]$). The first iteration corresponds to the worst case of the localization process (the biggest research domain).



(a)



(b)

Figure 8. Figure 8(a) represents the experimental environment. Figure 8(b) represents the considered map obtained by performing a SLAM algorithm. White cells represent free spaces in the environment, black cells represent obstacles and grey cells represent unexplored spaces. Note that the grey circle in Figure 8(b) corresponds to the pose of the robot depicted in Figure 8(a) and the filled points correspond to the 33 performed global localizations.

Table 2. Results of the 33 successful global localizations (data of the first iteration)

	Average	Best	Worst	Unity
Execution time	33.4	21	45.5	seconds
Precision of x_1	± 37.2	± 18.1	± 122.1	mm
Precision of x_2	± 32.7	± 16.5	± 51.5	mm
Precision of θ	± 2.75	± 0.55	± 12.2	degrees

4.3 *The CAROTTE arena*

The following data (map and sensor measurements) were recorded during the CAROTTE challenge in June 2011, by the CARTOMATIC team.

The considered environment, Figure 2.2, is a 20m $\times$ 20m indoor environment. The objective of the following experimentation is to test the IAL algorithm with a larger and a worse map than the previous experimentation. The considered map is depicted in Figure 9.

Even in those conditions the IAL algorithm is robust enough to provide a localization. Here are the results of a performed global localization (Figure 10):

- Execution time: 28s,
- Precision of x_1: ± 114.2mm,
- Precision of x_2: ± 105.8mm
- Precision of θ: $\pm 2, 87\,\text{deg}$

5. Comparison Between IAL and MCL

In this section the IAL algorithm is compared with a classic **M**onte **C**arlo **L**ocalization (MCL) algorithm.

5.1 *Monte Carlo Localization*

A probabilistic approach for the global localization problem is to compute a probability distribution over all the possible poses of the robot in the environment (Markov localization, [22]). This distribution, called $Bel(\mathbf{q}(k))$, expresses the robot's *belief* for being at the pose $\mathbf{q}(k)$. $Bel(\mathbf{q}(0))$ represents the initial state of knowledge. MCL method [23] represents this belief by maintaining a set of m particles $\mathbb{S}(k) = \{s_1(k), s_2(k), ..., s_m(k)\}$, drawn from it. A particle $s_i(k)$ is defined by

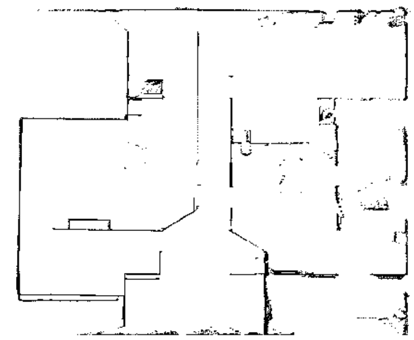


Figure 9. A map of the CAROTTE arena, obtained during the contest.

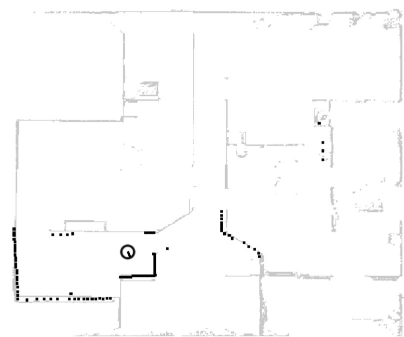


Figure 10. The result of a global localization. The grey cells correspond to the obstacles in the map and the black points correspond to the data set used to perform the localization. The black circle corresponds to a solution $\mathbf{q}(k) \in [\mathbf{q}(k)]$ of the global localization result.

its pose $\mathbf{q}_i(k) = (x_{1_i}(k), x_{2_i}(k), \theta_i(k))$ and a score $q_i(k)$ corresponding to the likelihood of obtaining the data set $\mathbf{y}(k)$ from the pose $\mathbf{q}_i(k)$. The initial belief $Bel(\mathbf{q}(0))$ is represented by a set of particles $\mathbb{S}(0)$ drawn according to an uniform distribution all over the considered environment.

To update the belief, two probabilistic models are used: a motion model to consider the control data and a perception model to exploit the sensor data. The robot motion is modelled by the conditional probability $p(\mathbf{q}(k)|\mathbf{q}(k-1), \mathbf{u}(k))$ and the sensor measurements by the probability $p(\mathbf{y}(k)|\mathbf{q}(k))$.

The MCL algorithm, Algorithm 2, has two important steps. First a re-sampling phase using the motion model, lines 2 and 3, that creates a new set of particles $\mathbb{S}(k)$ according to the previous step $\mathbb{S}(k-1)$ and the control vector $\mathbf{u}(k)$. The second step is the update of the new particle scores $q_i(k)$ using the data from the sensor $\mathbf{y}(k)$ and the sensor model.

5.2 *Computation time*

From the experimental results presented in Section 4, it appears that the first iterations of the IAL algorithm cost a large computation time.

On the other hand, when the research domain has been significantly reduced, the IAL algorithm computation time is comparable to the MCL algorithm.

In the following the computation time of a localization with a small research domain is tested. The experimentation considers a 10m×10m simulated environment (Figure 11), 36 measurements for each iteration (5 outliers), and a box $[\mathbf{q}(0)]$ with a size $1\text{m} \times 1\text{m} \times 10\,\text{deg}$. Here are the average

Algorithm 2: Monte Carlo Localization

Data: $\mathbb{S}(k-1), \mathbf{y}(k), \mathbf{u}(k)$

for $i = 1$ **to** m **do**

  generate random s from $\mathbb{S}(k-1)$ according to $q_1(k-1), ..., q_m(k-1)$;

  generate random $s' \backsim p(s'|s, \mathbf{u}(k))$;

  $q_{s'} = p(\mathbf{y}(k)|s')$;

  add $\langle s', q_{s'} \rangle$ to $\mathbb{S}(k)$;

normalize the importance factors $q_{s'}$ in $\mathbb{S}(k)$;

Result: $\mathbb{S}(k)$.

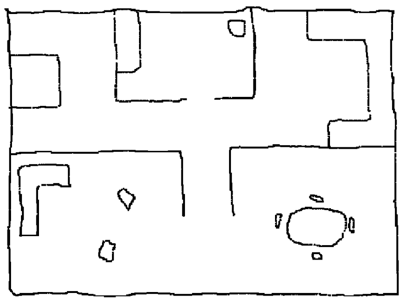


Figure 11. A simulated environment: the black pixels represent the grid map.

results of 13 successive localization iterations:

- MCL (5000 particles)
  - Execution time: 0, 07s.
- IAL
  - Execution time: 0, 06s.

It appears that in this case IAL and MCL have the same computation time.

When considering an entire exploration mission, the robot is more often localizing itself considering its previous pose than performing a global localization over all the environment. In fact, the worst case of the localization corresponds to the beginning of the mission and the recovering after each kidnapping.

Example: Considering an exploration mission of 4000s in the LORIA arena presented in the Section 4.2. Assuming that during this mission 4 kidnappings were performed, using the IAL method the mission time would be increased of $33.4\text{s} + 4 \times 33.4\text{s} = 167\text{s}$, with 33.4s the average localization time observed in this environment (Table 2). This increase corresponds to the cost of the first localization adding the four kidnapping recoveries. That would lead to a 4.2% increase of the computation time over the entire mission compared to a MCL approach. In those conditions the computation time of both methods are similar.

Furthermore, the IAL algorithm can process a static localization (without moving the robot), whereas the MCL algorithm uses the moving of the robot to avoid, for example, local minima. Thus the moving time of the robot should be added to the MCL computation time, reducing the computation time difference of the two algorithms.

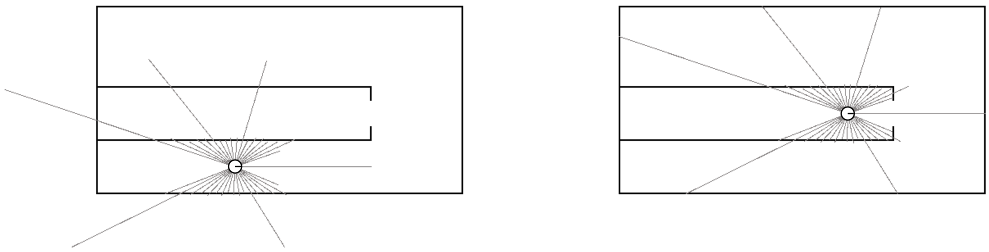
5.3 *Local/Global minima*

An obvious weakness of the Monte Carlo algorithm is that it can be stocked in a local minima and do not converge to the expected solution. For example it is the case when there is too few particles. In practice with enough particles the MCL manages to find the global minimum.

A question can be asked, however. Is the global minimum the solution of the localization problem? It is when considering a perfect world (perfect sensors, no outlier), but as soon as there are outliers in the data set, this assumption is not verified any more.

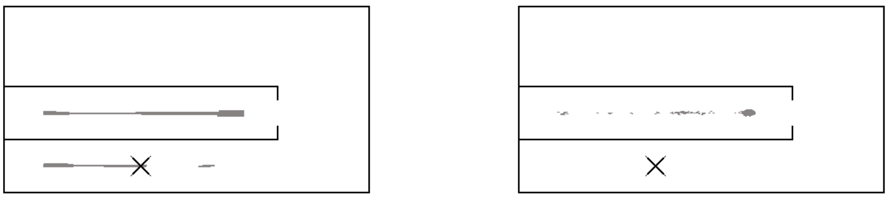
Figure 12(a) represents a simulated pose of a robot in an environment. As it can be seen 8 sensor measurements are outliers. In this case, the best possible score does not correspond to the robot's pose. In other words, the pose $\mathbf{q}_{best}(k)$ that maximises the probability $p(\mathbf{y}(k)|\mathbf{q}_{best}(k))$ is not the solution of the localization problem. For example Figure 12(b) represents a pose with a better score than the robot's pose.

In those conditions the MCL algorithm does not converge to the expected pose while the IAL algorithm still return a set that contains the solution (see Figures 13(b) and 13(a)). It exists MCL ameliorations with random sampling to recover from those kind of situations [24] but if this situation occurs during all the localization process, the MCL algorithm and its ameliorations will not converge to the actual localization.



(a) (b)

Figure 12. Figure 12(a) represents the considered map (black pixels), sensor data set (light grey pixels) and robot's pose (black circle). Figure 12(b) represents a pose with a better score than the expected one.



(a) (b)

Figure 13. Figure 13(a) represents the IAL results and Figure 13(b) the MCL results (the grey shapes). The crosses correspond to the expected result. Note that the best particle of the MCL algorithm is the one depicted in Figure 12(b). In those conditions MCL fails to estimate the robot's pose while the IAL algorithm results still contain the solution.

6. Conclusion

In this paper a set membership approach has been presented to deal with localization problems in mobile robotics. This method, combining several interval analysis tools appears to be efficient in a real context. The algorithm allows to perform a global localization and to handle kidnapping situations, even with outliers in the measurement set.

Then this method is compared to a MCL algorithm, which is mainly used to deal with the global localization problem. This comparison revels that interval analysis can be an efficient alternative to this probabilistic approach. Even if the first iteration of the IAL algorithm costs more computation time than a classical MCL, when considering an entire exploration mission the computation times of both algorithms are similar. Furthermore the proposed method is rigorous since it considers all the possible solutions regards to the localization problem and does not search a global minima that is assumed to be the solution.

All the presented experimentations have been processed on a distant computer, the next step of our work is to implement this algorithm in an actual robot in order to test the algorithm in real time.

Acknowledgements

This work has been partially supported by the French National Research Agency (Agence Nationale de la Recherche, ANR) and General Delegation of Armament (Direction Générale de l'Armement, DGA) through the Cartomatic project in ANR/DGA CAROTTE Challenge.

The data presented Section 4.2 were provided by A. Bautin and N. Beaufort, from the LORIA[1] laboratory.

References

[1] Borenstein J, Everett HR, Feng L. Navigating mobile robots: Systems and techniques. A. K. Peters, Ltd.. 1996.
[2] Cox IJ, Wilfong GT. Autonomous robot vehicles. . 1990.
[3] Choset H, Lynch KM, Hutchinson S, Kantor G, Burgard W, Kavraki LE, Thrun S. Principles of robot motion: Theory, algorithms, and implementations. MIT Press. 2005 June.
[4] Thrun S, Burgard W, Fox D. Probabilistic robotics. Intelligent robotics and autonomous agents. MIT Press. 2005.
[5] Arras K, Vestli S. Hybrid, high-precision localisation for the mail distributing mobile robot system mops. In: Robotics and automation, 1998. proceedings. 1998 ieee international conference on. Vol. 4. 1998 may. p. 3129 –3134 vol.4.
[6] Smith R, Self M, Cheeseman P. Estimating uncertain spatial relationships in robotics. Springer-Verlag New York, Inc.. 1990. p. 167–193.
[7] Jensfelt P, Kristensen S. Active global localization for a mobile robot using multiple hypothesis tracking. IEEE Transactions on Robotics. 2001;17(5):748–760.
[8] Leonard J, Whyte DH. Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robotics and Automation. 1991;7.
[9] Durrant-Whyte H, Majumder S, Thrun S, de Battista M, S Scheding S. A bayesian algorithm for simultaneous localisation and map building. In: Jarvis R, Zelinsky A, editors. Robotics research. Springer Tracts in Advanced Robotics. Springer Berlin / Heidelberg. 2003.
[10] Smith R, Cheeseman P. On the representation and estimation of spatial uncertainty. 1986.
[11] Cen G, Matsuhira N, Hirokawa J, Ogawa H, Hagiwara I. Mobile robot global localization using particle filters. In: International conference on control, automation and systems (iccas). 2008. p. 710–713.
[12] Dellaert F, Fox D, Burgard W, Thrun S. Monte carlo localization for mobile robots. Proceedings 1999 IEEE International Conference on Robotics and Automation (ICRA). 1999;128:1322–1328.
[13] Thrun S, Fox D, Burgard W, Dellaert F. Robust monte carlo localization for mobile robots. Artificial Intelligence. 2001;128(12):99 – 141.
[14] Zhang L, Zapata R, Lépinay P. Self-adaptive monte carlo localization for mobile robots using range sensors. In: Proceedings of the international conference on intelligent robots and systems (iros). 2009. p. 1541–1546.
[15] Jaulin L. A nonlinear set membership approach for the localization and map building of underwater robots. Robotics, IEEE Transactions on. 2009 feb;25(1):88 –98.
[16] Seignez E, Kieffer M, Lambert A, Walter E, Maurin T. Experimental vehicle localization by bounded-error state estimation using interval analysis. In: Intelligent robots and systems, 2005. (iros 2005). 2005 ieee/rsj international conference on. 2005 aug. p. 1084 – 1089.
[17] Ashokaraj I, Tsourdos A, Silson P, White B. Sensor based robot localisation and navigation: using

[1]Laboratoire Lorrain de Recherche en Informatique et ses Applications, Nancy (France)

interval analysis and extended kalman filter. In: Control conference, 2004. 5th asian. Vol. 2. 2004. p. 1086 –1093 Vol.2.
[18] Jaulin L, Kieffer M, Didrit O, Walter E. Applied interval analysis. Springer. 2001.
[19] Neumaier A. Interval methods for systems of equations (encyclopaedia of mathematics and its applications). Cambridge University Press. 1991.
[20] Rossi F, Beek P, Walsh T. Handbook of constraint programming (foundations of artificial intelligence). Elsevier Science Inc.. 2006.
[21] Kieffer M, Jaulin L, ric Walter, Meizel D. Robust autonomous robot localization using interval analysis. Reliable Computing. 2000;6:337–362.
[22] Cassandra A, Kaelbling L, Kurien J. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In: In proceedings of the ieee/rsj international conference on intelligent robots and systems. 1996. p. 963–972.
[23] Fox D, Burgard W, Dellaert F, Thrun S. Monte carlo localization: Efficient position estimation for mobile robots. In: Aaai/IAAI. 1999. p. 343–349.
[24] Lenser S, Veloso M. Sensor resetting localization for poorly modelled mobile robots. In: In proceedings of the ieee international conference on robotics and automation (icra). IEEE. 2000.